



INTÉGRATION DES AFFICHEURS DITEL SÉRIE DINOS-DMG DANS LES SYSTÈMES OPÉRATIFS WINDOWS 32 ET 64 BITS

INDEX

1. Configuration minimale requise du PC

MPComm est un composant ActiveX. Ce composant peut être utilisé avec les sys tèmes opératifs Windows 32 et 64 bits.

La technologie ActiveX peut être utilisée avec la plupart des systèmes actuels de développement, de manière que **MPComm**.

Les tests ont été effectués en utilisant C# et VisualBasic.

Pour le bon fonctionnement de **MPCOMM** il est nécessaire d'établir les trois pro priétés relatives à l'octroi de la licence. Voir exemple dans le chapitre suivant.

LicProduct	Nom du Produit	**MPCOMM**
LicName	Nom de l´Entreprise	**PIXELMP**
LicKey	Code de licence	**“431A2CF1683AAFAC”**

2. Fonctionnement général

MPComm est un composant ActiveX qui peut être intégré dans n'importe quelle application 32 bits développée avec les derniers systèmes de développement.

Le contrôle comprend:

Propiétés : Elles nous permettent de configurer et contrôler les ports série RS232, les connexions par le protocole TCP / IP, définir et récupérer l'information au cours des pro cessus de communication entre le contrôle et les différents appareils à contrôler.

Événnements : Les événements nous informent en temps d´exécution de l'activité du contrôle avec les différents appareils à contrôler.

Méthodes : Les méthodes nous permettent d'exécuter des fonctions propres du contrôle.

Pour utiliser le contrôle il faut d´abords initialiser le composant MPComm puis ouvrir le port série ou établir une connexion TCP / IP, on pourra alors exécuter le contrôle souhaité. Après chaque contrôle il est recommandé fermer la communication.

Exemple : initialiser MPComm (C#)

```
axMPComm1.LicProduct = "MPCOMM";          //Nom du composant
axMPComm1.LicName = "PIXELMP";            //Nom du composant
axMPComm1.LicKey = "431A2CF1683AAFAC";    //Code licence
axMPComm1.Terminal2 = 8;                  //
axMPComm1.Terminal1 = 4;                  //
axMPComm1.DebugMode = false;              //Mode de dépuration
```

Exemple : ouvrir Port Série (VB)

```
MPComm1.Port = "COM1"   // Port
MPComm1.Speed = "38400" // Vitesse
MPComm1.Protocolo = 0   // Protocole.   0 -> RS232/485, 1 -> TCP/IP
MPComm1.RS485 = false   // Type ligne. False ->RS232, True -> RS485
MPComm1.OpenComm        // Ouvrir port
```

Exemple : ouvrir Port Série (C#)

```
    axMPComm1.Port = "COM1:";
    axMPComm1.Speed = "9600";
    axMPComm1.Protocolo = 0;
    axMPComm1.RS485 = false;
    axMPComm1.OpenComm();
```

Exemple : établir communication TCP / IP (VB)

```
MPComm1.HostAdress = gl_iphost$  //IP Afficheur
MPComm1.HostPort = gl_ipport$  // TCP port
MPComm1.Usuario = gl_ipusr$      //
MPComm1.Password = gl_ippass$
MPComm1.Protocolo = 1
MPComm1.RS485 = False
MPComm1.OpenComm
```

Exemple : établir communication TCP / IP (C#)

```
axMPComm1.TipoServidor = 1;    //  Type serveur.Valeur par défaut
    axMPComm1.HostPort = 53;                //  Port TCP.Valeur par défaut
    axMPComm1.Usuario = "admin";     //  Usager.Valeur par défaut
axMPComm1.Password = "security";    //  Password .Valeur par défaut
axMPComm1.HostAdress = "192.168.1.44"; //Adresse  IP afficheur
axMPComm1.Protocolo = 1;              //Protocole.   0 -> RS232/485, 1 -> TCP/IP
axMPComm1.OpenComm();              // Ouvrir port
```

Après l'ouverture d'une communication on peut communiquer avec les différents appareils. Dans l'exemple suivant, nous transmettons à l'afficheur 1 un programme de test intitulé «**Test1**» en mode **IMMÉDIAT** en utilisant la propriété **Orden** (voir page 18) á laquelle on assigne la valeur 39 (0x27) qui corresponds á la commande **FASTEXEC** (exécution immédiate d´un message).La liste détaillée des commandes que l´on peut utiliser avec la propriété **Orden** se trouve à la page 21.

Exemple : Envoyer un message à l´afficheur (VB)

```
MSG$ = Chr$(MPMOD_INMEDIATO) + "Test1"   // message à envoyer (String)
MPComm1.BufferTX = MSG$                       // Buffer = message
MPComm1.Orden = 0x27        // commande FASTEXE (Envoyer inmédiatement)
numero_error = Str$( MPComm1.ErrorNx     // résultat communication
Error$ = MPComm1.ErrorTx                    // Texte Erreur
```

Exemple : Envoyer un message à l´afficheur (C#)

```
string MSG = "Test1";                  // message à envoyer (String)
axMPComm1.Pantalla = 1;                 // Afficheur avec ID nº 1
axMPComm1.BufferTX = MSG;              // Buffer = message
axMPComm1.Orden = 0x27;      // commande FASTEXE (Envoyer inmédiatement)
Thread.Sleep(200);                    // Temporisation de 200ms
String s = axMPComm1.        // Réponse de l´afficheur(si retour valeur)
int error = axMPComm1.ErrorNx;          // Résultat de la communication
```

Il est important de différencier les commandes de communication (FASTEXEC, EXECUTE, SETHORA) des commandes de programmation (INMEDIATO,SUBE,VELMODO). Les commandes de communication nous permettent d'envoyer et de recevoir des informations d´un afficheur. Les commandes de programmation nous permettent d'établir une séquence d'instructions à exécuter sur l'afficheur une fois le programme transmit.

Enfin, vous devez fermer la connexion pour mettre fin aux processus de communication.

Exemple : Terminer la communication (C#)

```
MPComm1.CloseComm   //Ferme le port série ou la connexion TCP/IP
```

3. Propriétés

 **BufferRx**

La propriété BufferRx nous permet d´obtenir le résultat de toutes les commandes qui renvoient des valeurs, comme par exemple, demander la date de l´afficheur, obtenir le programme en cours, le répertoire, etc.

Type	Lecture / Écriture	Format	Plage
String	Oui / Non	(Selon Commande)	-

```
MPComm1.Pantalla=1
MPComm1.Orden = 0x0B  // commande GETHORA (Demande
                           l'heure de l'afficheur)
Heure_Afficheur$ = MPComm1.BufferRx
```

 **BufferTx**

Certaines commandes du protocole ne sont pas limitées à réaliser une opération spécifique, mais permettent de transmettre des informations à l'afficheur. Cette information qui doit être transmise doit être stockée dans cette propriété avec le format adéquat de la commande avant d'exécuter la commande.

Tiye	Lecture / Écriture	Format	Plage
String	Non / Oui	(Selon Commande)	-

```
MPComm1.Pantalla = 1
MPComm1.BufferTx ="01-01-99 12:00:00"
MPComm1.Orden = 0x0A  // commande SETHORA (Met à
                 l´heure l´horloge de l´afficheur)
```

 **CFGNumPan** UNIQUEMENT POUR RS232 .

Définit le numéro de l´afficheur (ID) dans la configuration interne. Ce champ a une plage de 0 à 253.

Type	Lecture / Écriture	Format	Plage
Entier	Si/Si	-	0 - Non utilisé 1–253 Recommandé 254 - Maître 255 - Diffusion

```
MPComm1.CFGNumPan = 1
```

 **CFGBaudsRS232** Uniquement pour RS232

Indique la vitesse du port RS232 de la carte de contrôle dans la configuration interne. Une fois modifier ce paramètre dans la configuration on doit fermer le port de communication pour modifier les paramètres et l'ouvrir à nouveau.

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui		0-1200 1-2400 2-4800 3-9600

```
MPComm1.CFGBaudRS232 = 3
MPComm1.Orden = 0x0B  // comando GETHORA (Pide la ho
ra del visualizador)
Hora_Visualizador$ = MPComm1.BufferRx
```

 **CFGPassword**

Cette propriété permet de modifier ou supprimer le mot de passe sur la carte de contrôle. Si on définit cette propriété avec huit espaces et on transmet, ce paramètre est désactivé. Pour s'assurer que les changements de mots de passe ont été réalisés vous devez redémarrer l'afficheur.

Type	Lecturae/ Écritur	Format	Plage
String	Oui / Oui	8A	-

```
MPComm1.CFGPassword = “CSECRETO”
```

 **CFGBaudsRS485**

Spécifie la vitesse de la configuration interne du port RS485. Si la carte de contrôle ne dispose pas du port RS485 , elle doit correspondre à la vitesse du port RS232. Dans le cas contraire les deux ports peuvent avoir des vitesses différentes..

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui		0-1200 1-2400 2-4800 3-9600

```
MPComm1.CFGBaudsRS485 = 1
```

 **CFGOffsetTemp**

Cette propriété détermine la configuration de la carte de contrôle dont la valeur devrait augmenter ou diminuer l'affichage de la température. Le champ n'affecte pas si aucune sonde de température ou de contrôle est installée.

Type	Lecture / Écriture	Format	Plage
Signed	Oui / Oui	-	-

MPComm1.CFGOffsetTemp = -1

 **CFGFlPowerOnReset**

Indiquez si la carte de contrôle au début devrait être remis à zéro automatiquement et initialiser la mémoire d'affichage.

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui		0-Oui 1-No

MPComm1.CFGFlPowerOnReset = 1

 **CFGFlNoCls**

Le flag de ClsStop est utile car elle nous permet de définir si on souhaite lors d´un STOP de l'afficheur, effacer également le message. Dans certains cas, l'arrivée continue de l'information provoque scintillement de l'afficheur, cela peut être évité avec cette propriété.

NOTE :Cette Même action peut se réaliser simplement en envoyant un message vide avant d´arrêter l´afficheur.

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui		0-Oui 1-No

MPComm1.CFGFlNoCls = 1

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	-	0-255

MPComm1.CFGVelModoDef = 1



DebugMode

Cette propriété offre la possibilité de générer un registre à l'emplacement C: \ MPCOMM.DBG avec toutes les communications, indiquant la direction et le nombre de millisecondes entre les communications. Cette propriété est utile pour déboguer le produit final.

Type	Lecture / Écriture	Format	Plage
Bool	Oui / Oui	-	True-False

```
2768895 --> (1)16 15 00 01 13 42 42 56 00 B0 40 00 00 04 C9 00 00 30 39 3F 03
2768970 <-- ACK:00
2872454 --> (1)16 0F 00 01 06 49 4E 46 4F 42 4F 4C 53 88 02
2872595 <-- ACK:00
3788235 --- Puerto serie Cerrado
```



ErrorNx

Utilise la propriété ErrorNx après chaque appel pour vérifier le résultat de celui-ci. Normalement, si la communication a réussi La propriété aura la valeur 0.Utile pour définir si la communication a réussie et en cas contraire montrer le message d´erreur (ErrorTX)

Type	Lecture / Écriture	Format	Plage
Entier	Oui / No	-	0-1050

```
Dim Numero_error AS Integer

MPComm1.Pantalla = 1
MPComm1.BufferTx =chr$(MPMODO_INMEDIATO)+"PRUEBA DE TEXTO"
MPComm1.Orden = MPCMD_FASTEXEC

Numero_error = MPComm1.errornx
```



ErrorTx

La propriété ErrorTx est très similaire à ErrorNx avec une particularité très importante. L'erreur ci-dessus est retournée comme une chaîne. Cette propriété est mise à jour après chaque communication.

Type	Lecture / Écriture	Format	Plage
String	Oui / No	-	-

```
Dim Numero_error AS
Integer
Dim Texto_error AS String
MPComm1.Pantalla = 1
MPComm1.BufferTx =chr$(MPMODO_INMEDIATO)+"PRUEBA DE TEXTO"
MPComm1.Orden = MP_FASTEXEC
if MPComm1.errornx<>0 then
Texto_error=MPComm1.ErrorTx
Msgbox (Texto_error)
end if
```



EsperaACK

Spécifie le temps en millisecondes à attendre avant de donner un ACK erreur de timeout dispositif

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	(IP)	1-10000

MPComm1.EsperaACK = 3000 (3 secondes)



HostAdress

Propriété HostAdress est responsable pour stocker l'adresse IP de l´afficheur.

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	(IP)	255.255.255.255

MPComm1.HostAdress=”127.0.0.1”



HostPort

Hostport fonctionne en conjonction avec HostAdress pour déterminer le port TCP de l´afficheur serveur auquel se connecter. Habituellement l'adresse IP et le numéro de port d'un serveur ne dépend pas de notre logiciel, mais nous aurons besoin d’avoir cette information.

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	-	1-65535

MPComm1.HostAdress=”127.0.0.1”
MPComm1.Hostport=”1001”



Orden

On utilise cette propriété pour indiquer au composant MPComm quel commande exécuter sur l'afficheur. Selon la commande le contrôle peux attendre ou pas la réponse de l'afficheur, et stocker dans **BufferRx** les données reçues ou bien utiliser le **BufferTx** pour transmettre des informations.

Type	Lecture / Écriture	Format	Plage
Entier	No / Oui	-	0-255

MPComm1.Pantalla = 1
MPComm1.BufferTx =chr$(MPMODO_INMEDIATO)+”PRUEBA DE TEXTO”
MPComm1.Orden = MP_FASTEXEC
MPComm1.Pantalla = 1
MPComm1.Orden = MP_GETDIR
Directorio$= MPComm1.BufferRx

 **Pantalla**

La propriété Pantalla définie l'adresse ID de l´afficheur avec lequel on veut réaliser le processus de communication. L'afficheur 255 est réservé à la diffusion Broadcast et dans ce cas aucune réponse n'est attendue .

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	-	0-255

MPComm1.Pantalla = 1

 **PassWord**

La propriété PassWord nous permet de définir le mot de passe à utiliser lors d'une communication avec le protocole TCP / IP. Cette propriété est conjointe à la propriété Usuario

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	-	0-10

MPComm1.PassWord ="ULZ2300"

 **Port**

La propriété Port stocke le numéro de port du port RS232 pour être ouvert et utilisé pour les transmissions. Cette propriété est utilisée lorsque vous travaillez avec le protocole RS232.

Type	Lecture / Écriture	Format	Plage
Entier	No / Oui	-	COM1 - COM4

MPComm1.Port="COM1"

 **Protocolo**

Cette propriété est très importante car elle permet la commutation du protocole ou support de transmission entre RS232 et TCP / IP. Le mode ne peut pas être changé une fois le port a été ouvert.

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	-	0-1

MPComm1.Protocolo = MPPROT_RS232

0 Protocolo RS232
1 Protocolo TCP/IP



Reintentos

Cette propriété nous permet de définir le nombre de tentatives qui souhaitent effectuer avant d'envisager une éventuelle erreur. Dans la plupart des cas, trois tentatives sont habituellement effectuées, mais dans certains cas il peut être préférable d'utiliser une seule.

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	-	0-10

MPComm1.Reintentos = 1



Speed

Propriété Speed est utilisé pour définir la vitesse de communication entre le PC et les afficheurs d'affichage ou d'un réseau RS485. Les vitesses disponibles sont

Type	Lecture / Écriture	Format	Plage
String	Oui / Oui	-	1200 2400 9600 19200 38400

```
MPComm1.Speed = ”9600”
MPComm1.Port    = ”COM1”
MPComm1.OpenComm
```



Usuario

Définit le nom d'utilisateur à utiliser pour établir une connexion avec l´afficheur. Cette propriété peut être utilisée que lorsque l´on utilisez le protocole TCP/ IP et son utilisation est conjointe à la propriété PassWord

Type	Lecture / Écriture	Format	Plage
String	Oui / Oui	-	-

MPComm1.Usuario = ”Administrador”

 **VariableIDX**

Lorsque l´on souhaite utiliser les variables numériques internes de l´afficheur cette propriété définit le code de la variable à modifier. Les variables sont identifiées numériquement, avec la variable A égale au numéro 0.

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	-	1-28

```
MPCommObj.VariableIdx = 2  ‘ Variable C
MPCommObj.VariableValor = Val(Text1(0))
MPCommObj.Orden = MPCMD_PUTVARS
```

 **VariableValor**

Définit le format de la variable qui a été sélectionné avec VariableIDx. Cette valeur est entrée en format entier. On indique le format de sortie sur l´afficheur en indiquant le nombre de chiffres et le nombre de décimales devant le nom de la variable.

Exemple : 5.2A 5 chiffres avec 2 décimales pour la variable A

Type	Lecture / Écriture	Format	Plage
Entier	Oui / Oui	-	1-65535

```
MPCommObj.VariableIdx = 2  ‘ Variable C
MPCommObj.VariableValor = 123
MPCommObj.Orden = MPCMD_PUTVARS
```

Exemple de programme avec des variables. Les variables peuvent se définir

```
txt$ = Chr$(MPDAT_SYNC)
txt$ = txt$ + Chr$(MPDAT_LINIA) + "1"
txt$ = txt$ + Chr$(MPMOD_INMEDIATO) + " VAR A=" + Chr$(MPTIM_VARIABLE) + "A"
txt$ = txt$ + Chr$(MPDAT_LINIA) + "2"
txt$ = txt$ + Chr$(MPMOD_INMEDIATO) + "VAR B=" + Chr$(MPTIM_VARIABLE) + "5.2B"
txt$ = txt$ + Chr$(MPDAT_LINIA) + "3"
txt$ = txt$ + Chr$(MPMOD_INMEDIATO) + "VAR C=" + Chr$(MPTIM_VARIABLE) + "5.0C"
txt$ = txt$ + Chr$(MPDAT_SYNC)

MPCommObj.OpenComm
If MPCommObj.ErrorNx Then Exit Sub


MPCommObj.Pantalla = 1
MPCommObj.BufferTX = Chr$(MPMOD_INMEDIATO) & txt$
MPCommObj.Orden = MPCMD_FASTEXEC
```

---

4. Méthodes

 **OpenComm**

Ouvrez le port série ou un protocole TCP / IP en utilisant les propriétés relatives au protocole, la vitesse et le port série. Cette méthode n'utilise pas de paramètres. Le résultat de la méthode se déplace a la propriété ErrorTX et ErrorNX.

Parámetros
-

```
MPComm1.Speed = ”9600”
MPComm1.Port    = ”COM1”
MPComm1.OpenComm
```

```
MPComm1.HostAdress = “195.106.100.2”
MPComm1.HostPort = “1001”
MPComm1.Usuario = “Usuario”
MPComm1.Password = “Clave”
MPComm1.Protocolo = 1
MPComm1.OpenComm
```

 **CloseComm**

Ferme le port série ou une communication TCP / IP . Cette méthode n'utilise pas de paramètres. Le résultat de la méthode se déplace a la propriété ErrorTX et ErrorNX.

Parámetros
-

```
MPComm1.CloseComm
```

---

5. Détails d´utilisation des commandes.

```
MPCMD_RESET = 1          ' Initialise afficheur

MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = “COM1”
MPComm1.Speed = “9600”
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.Orden = MPCMD_RESET
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm

MPCMD_WDRESET = 2        ' Réinitialise l'afficheur
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = “COM1”
MPComm1.Speed = “9600”
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.Orden = MPCMD_WDRESET
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm

MPCMD_STOP = 3           ' Arrête l'exécution de l'afficheur

MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = “COM1”
MPComm1.Speed = “9600”
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.Orden = MPCMD_STOP
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

```
MPCMD_BORRAR = 5        ' Supprime un programme
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.BufferTX = "RELOJ"
MPComm1.Orden = MPCMD_BORRAR
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

MPCMD_GETPROG = 9 ' Demande le code du programme en train de s´exécuter dans l´afficheur. // Par exemple pour récupérer le code d´un programme fait avec le software d´édition Dynamic Plus.Le code récupéré peut se réutiliser directement avec la commande MPCMD_FASTEXEC

```
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.BufferTX = "PRUEBA"
MPComm1.Orden = MPCMD_GETPROG
Programa$ = MPComm1.BufferRx
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

```
MPCMD_SETHORA = 10      ' Établit l'heure de l'affichage
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.BufferTX = "01-01-05 13:30:16"
MPComm1.Orden = MPCMD_SETHORA
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

MPCMD_GETHORA = 11 ' Demande l'heure de l'affichage

```
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.Orden = MPCMD_GETHORA
iret$ = MPComm1.BufferRx

        TxtFecha(2) = Asc(Mid(iret$, 1, 1))
        TxtFecha(1) = Asc(Mid(iret$, 2, 1))
        TxtFecha(0) = Asc(Mid(iret$, 3, 1))

        TxtHora(0) = Asc(Mid(iret$, 4, 1))
        TxtHora(1) = Asc(Mid(iret$, 5, 1))
        TxtHora(2) = Asc(Mid(iret$, 6, 1))

ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

MPCMD_PASSWORD = 14 ' Envoyer le password a l'afficheur

```
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.BufferTX = "SECRET"
MPComm1.Orden = MPCMD_PASSWORD
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

```
MPCMD_SETPASSW = 15     ' Établit nouveau password
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.BufferTX = "SECRET"
MPComm1.Orden = MPCMD_SETPASSW
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm


MPCMD_GETVER = 18       ' Demande de version du logiciel
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.Orden = MPCMD_GETVER
respuesta$ = main.MPComm1.BufferRx
version = Val("&h" + Mid(respuesta$, 1, 2))
hardhare$ = Mid(respuesta$, 4)
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm


MPCMD_NGETDIR = 30      ' Demander le répertoire à l'afficheur
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Orden = MPCMD_NGETDIR
Directorio$   =MPComm1.BufferRx
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

MPCMD_NEJECUTAR = 31 ' Exécuter un programme sur l'afficheur

```
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.BufferTX = "MPTEST"
MPComm1.Orden = MPCMD_NEJECUTAR
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

MPCMD_FASTEXEC = 39 ' Exécution immédiate d'un programme

```
MPComm1.DebugMode = false
MPComm1.Terminal2 = 8
MPComm1.Terminal1 = 4
MPComm1.Port = "COM1"
MPComm1.Speed = "9600"
MPComm1.Protocolo = 0
MPComm1.RS485 = False
MPComm1.OpenComm
MPComm1.Pantalla = 1
MPComm1.BufferTX = Chr$(MPMOD_INMEDIATO) +
"Prueba"
MPComm1.Orden = MPCMD_FASTEXEC
ErrorNx  = Str$(MPComm1.ErrorNx)
ErrorTx$ = MPComm1.ErrorTx
MPComm1.CloseComm
```

6. Liste des commandes

À définir dans l´application comme variables globales en format « string».

```
Ejemplo (C#) :
string MPCMD_FASTEXEC = char (0x27) +"";
axMPComm1.Orden = MPCMD_FASTEXEC;
```

```
Global Const MPCMD_RESET = 1(0x01)' Initialise l´afficheur
Global Const MPCMD_WDRESET = 2(0x02)' Réinitialise l'afficheur
Global Const MPCMD_STOP = 3(0x03)' Arrête l'exécution de l'afficheur
Global Const MPCMD_BORRAR = 5(0x05)' Supprime un programme
Global Const MPCMD_CONFIG = 8(0x08)' Configuration des ports d'afficheur
Global Const MPCMD_GETPROG = 9(0x09)' Demande le code du programme en train
                                      de s´exécuter dans l´afficheur.
Global Const MPCMD_SETHORA = 10(0x0A)' Met à l´heure l´horloge de l´afficheur
Global Const MPCMD_GETHORA = 11(0x0B)' Demande l'heure de l'afficheur
Global Const MPCMD_PASSWORD = 14(0x0E)' Envoyer le mot de passe à l'afficheur
Global Const MPCMD_SETPASSW = 15(0x0F)' Définit nouveau mot de passe
Global Const MPCMD_SETTEMP = 16(0x10)' Etablie offset température
Global Const MPCMD_GETTEMP = 17(0x11)' Demande température
Global Const MPCMD_GETVER = 18(0x12)' Demande  version du logiciel
Global Const MPCMD_NGETDIR = 30(0x1E)' Demander le répertoire à l'afficheur
Global Const MPCMD_NEJECUTAR = 31(0x1F)' Exécuter un programme sur
                                         l'afficheur.Voir software d´édition
                                         Dynamic Plus.
Global Const MPCMD_FASTEXEC = 39(0x27)' Exécution immédiate d'un programme
```

7.ANEXO B Protocolos soportados

```
Global Const MPPROT_RS232 = 0   ' Mode de transmission pour série
Global Const MPPROT_TCPIP = 1   ' Mode de transmission pour Sockets TCP/IP
```

8.ANEXO C   Constantes para programas

```
Global Const MPTIM_DIF_DIAS = &HA4
Global Const MPTIM_DIF_SEMAN = &HA5
Global Const MPTIM_DIF_MESES = &HA6
Global Const MPTIM_HORMIN = &HA7
Global Const MPTIM_TEMPER = &HA8
Global Const MPTIM_DIAV = &HA9
Global Const MPTIM_MESV = &HAA
Global Const MPEFE_FLASH = &HB0
Global Const MPEFE_NEGATIVO = &HB1
Global Const MPEFE_BORRADO = &HB2
Global Const MPEFE_ESPERA = &HB3
Global Const MPEFE_BEEP = &HB4
```

```
Global Const MPDAT_GROSOR = &HC0
Global Const MPDAT_TIPO_LETRA = &HC1
Global Const MPDAT_INVERSO = &HC2
Global Const MPDAT_NORMAL = &HC3
Global Const MPDAT_VEL_MODO = &HC4
Global Const MPDAT_ESP_MODO = &HC5
Global Const MPDAT_CICLOS = &HC6
Global Const MPDAT_LINIA = &HC7
Global Const MPDAT_PROGRAM = &HC8
Global Const MPDAT_SYNC = &HC9
Global Const MPDAT_NOSYNC = &HCA
Global Const MPDAT_NOCENTRO = &HCD
Global Const MPDAT_ANIM = &HCE
Global Const MPDAT_NOANIM = &HCF
Global Const MPDAT_LUMIN = &HD0
Global Const MPDAT_REM = &HD2
Global Const MPDAT_DIBUJO = &HD4
Global Const MPDAT_VENTANA = &HD3
Global Const MPDAT_MPBASIC = &HD5
Global Const MPTDAT_BLINK = &HA0
Global Const MPTDAT_COLOR = &HA1
Global Const MPTDAT_FONDO = &HA2
Global Const MPTDAT_GRAFICO = &HA3
Global Const MPMOD_CORRER = &HE0
Global Const MPMOD_CENTRO = &HE1
Global Const MPMOD_DISMINUIDO = &HE2
Global Const MPMOD_APILADO = &HE3
Global Const MPMOD_RODAR = &HE4
Global Const MPMOD_SUBE = &HE5
Global Const MPMOD_BAJA = &HE6
Global Const MPMOD_ROTACION1 = &HE7
Global Const MPMOD_ROTACION2 = &HE8
Global Const MPMOD_ROTACION3 = &HE9
Global Const MPMOD_ROTACION4 = &HEA
Global Const MPMOD_APARICION1 = &HEB
Global Const MPMOD_APARICION2 = &HEC
Global Const MPMOD_APARICION3 = &HED
Global Const MPMOD_APARICION4 = &HEE
Global Const MPMOD_NIEVE = &HEF
Global Const MPMOD_INMEDIATO = &HF0
Global Const MPMOD_DESLIZAR = &HF1
Global Const MPMOD_GIRAR = &HF2
Global Const MPMOD_BOLSA = &HF3
```

9.ANNEXE D Liste des Méthodes,Propriété et évenements

La liste suivante est exhaustive et inclue également les propriétés, méthodes ou événements utilisés exclusivement para la programmation interne de l´afficheur en usine.

Ceux-ci n´apparaissent pas dans ce manuel qui est orienté à l´usager final .

Il est recommandé de ne pas modifier leur valeur par défaut et d´utiliser seulement ceux décrits dans ce manuel.

AboutBox()	TransmitTxt(string)	CFGFIModem
AddParametterShort(short)	TransmitWav(string)	CFGFINoCls
AddParametterString(string, shc	TransmitWavEx(strin	CFGFIPowerOnRes
AnimCrc(string)	UpdateLight(string, s	CFGFIRS485
AttachInterfaces()	AbortComm	CFGFontDef
AxMPComm()	BinaryTx	CFGGPSActiu
BufferDisk()	blubrillo	CFGGPSChangeAu
CloseComm()	blucontrast	CFGIdioma
CompressGMP(string, string, sh	BufferRx	CFGLastUpdate
CompressGMPEstimateLen(strir	BufferTX	CFGLinLen
CreateSink()	CFGAltLen	CFGLocalCast
DetachSink()	CFGAltLins	CFGLumDef
get_variant(short)	CFGBaudsRS232	CFGLumOutMax
GetLSNumNode()	CFGBaudsRS485	CFGLumOutMin
GetLSRdNode(short)	CFGCReceptor1	CFGModelo
GetStatus2Error(short)	CFGCReceptor10	CFGNSerial
GetStatus2Event(short)	CFGCReceptor11	CFGNumFa
GetStatus2EventEx(short)	CFGCReceptor12	CFGNumLin
GetStatus2Noms(short)	CFGCReceptor13	CFGNumPan
GetStatusError(short)	CFGCReceptor14	CFGOffsetInvierno
GetStatusEvent(short)	CFGCReceptor15	CFGOffsetTemp
GetStatusEventEx(short)	CFGCReceptor16	CFGOffsetVerano
GetStatusNoms(short)	CFGCReceptor2	CFGPassword
GetStatusSDACode()	CFGCReceptor3	CFGPixLins
GetStatusSDAMode()	CFGCReceptor4	CFGReceptor1
ImTickGetBitmap(string)	CFGCReceptor5	CFGReceptor10
ImTickGetFonts(string)	CFGCReceptor6	CFGReceptor11
InitParametters()	CFGCReceptor7	CFGReceptor12
ModemCall(string, string, int)	CFGCReceptor8	CFGReceptor13
ModemDCD()	CFGCReceptor9	CFGReceptor14
ModemHungUp(string, string)	CFGDedActPas	CFGReceptor15
OpenComm()	CFGDedCtrl	CFGReceptor16
OpenCommNonBlocking()	CFGDedDcf	CFGReceptor2
set_variant(short, object)	CFGDedEntCnt	CFGReceptor3
SetLSNumNode(int)	CFGDedFlcNiv	CFGReceptor4
SetLSRdNode(short, int)	CFGDedProgHeap	CFGReceptor5
SetSlaveCmd(short, short)	CFGDedSubBaj	CFGReceptor6
SetSubOrden(short)	CFGDispInv	CFGReceptor7
TransmitEditFilex(short, string)	CFGEfectoDef	CFGReceptor8
TransmitFont(string)	CFGEsperaModoDef	CFGReceptor9
TransmitGraf(string)	CFGFactUpdate	CFGRomOffset
TransmitGrafComp(int)	CFGFIColor	CFGSdaCode
TransmitSoft(string)	CFGFIIR	CFGSdaDias

CFGSdaOrg
CFGSenLum
CFGSPorts
CFGTimeOut
CFGTipoHard
CFGTipusP
CFGTmpMed
CFGVelModoDef
DebugMode
Dominio
ErrorDTP
ErrorNx
ErrorTx
EsperaAck
FileTmp
GetCntContador
GetCntValor
grebrillo
grecontrast
HostAdress
HostPort
ispriority
LicKey
LicName
LicProduct
Orden
PakIncSize
PakMaxSize
PakMinSize
Pantalla
PantallaControlLoc
Password
Port
Protocolo
PutCntContador
PutCntValor
redbrillo
redcontrast
Reintentos
RS485
RXFonLen
scoded
SelectedName

PakIncSize
PakMaxSize
PakMinSize
Pantalla
PantallaControlLock
Password
Port
Protocolo
PutCntContador
PutCntValor
redbrillo
redcontrast
Reintentos
RS485
RXFonLen
scoded
SelectedName
SelectedParentNode
Speed
StrCode1
StrCode2
StrCode3
StrCodeOp
StrIDPantalla
StrIndex
StrType
Terminal1
Terminal2
TickerCodigo
TickerDecimales
TickerPrecio
TickerTendencia
TickerVolumen
TipoGetHora
TipoHardware
TipoServidor
Usuario
VariableIdx
VariableValor
IDle
StatComExt
StatComm

10. Utilisation de MPCOMM.OCX dans une application .NET

Ce qui suit explique la meilleure méthode pour utiliser le composant MPCOMM.OCX dans une Applications.Net

Tous les exemples sont présentés dans le langage C #.

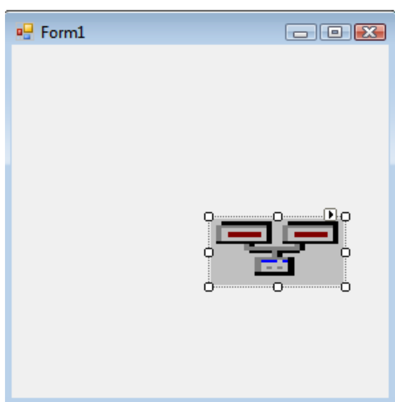
Créer des Wrappers COM (Component Object Model) pour l'OCX

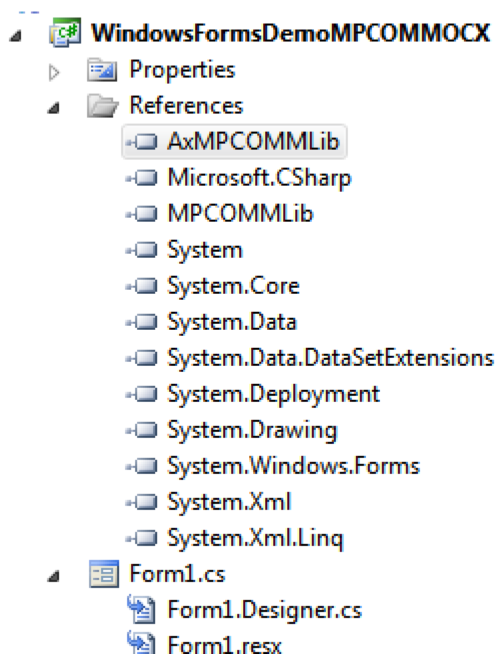
L'OCX ne peut pas être utilisé directement dans l'application.Il est nécessaire de créer une série de DLL (wrappers).
Le mode le plus simple est (pour Visual Studio 2010 et langage C #):

- Créer un formulaire, cliquez-droit sur la boîte à outils et sélectionnez "Choisir éléments".
- Cliquez sur l´onglet "Composants COM" .
- Localisez votre contrôle ActiveX (MPCOMM: OCX), cochez la case et cliquez sur "OK".
- Maintenant, il devrait apparaître sur la boîte à outils; faites glisser et déposez-le sur votre formulaire. Attribuez un nom (par exemple: AxMPCOMM1)

REMARQUE: vous verrez dans la section «références» de votre projet que 2 éléments ont été ajoutés automatiquement (wrappers):

. AxMPCOMMLib
. MPCOMMLib





11. Programming codes

Using the component

Configure Component

```
axMPComm1.LicProduct = "MPCOMM";
axMPComm1.LicName = "PIXELMP";
axMPComm1.LicKey = "431A2CF1683AAFAC";
axMPComm1.Terminal2 = 8;
axMPComm1.Terminal1 = 4;
axMPComm1.DebugMode = true;
```

Open Communications (in this case RS232):

```
// Open RS232
axMPComm1.Port = "COM1:";
axMPComm1.Speed = "9600";
axMPComm1.Protocolo = 0;     // 0 -> RS232/485, 1 -> TCP/IP
axMPComm1.RS485 = false;
```

Prepare Command to Send

```
// Open Comm
bool ret =axMPComm1.OpenComm();

// RS232 successfully open
if (ret)
{
    // Target device
    axMPComm1.Pantalla = 1;    // Device with ID 1

     // Mode 'Scroll' + 'MP test'
          axMPComm1.BufferTX = (char) 0xE0 + "MP Test";
          axMPComm1.Orden = 0x27;    // Send

            // Receive (if any)
            String s = axMPComm1.BufferRx;   // Get answer (if any)

            // Error
            int error = axMPComm1.ErrorNx;   // Get error number

            // Close Comm
            axMPComm1.CloseComm();
        }
```

Example 1: serial network

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsDemoMPCOMMOCX
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            // Configure component
            axMPComm1.LicProduct = "MPCOMM";
            axMPComm1.LicName = "PIXELMP";
            axMPComm1.LicKey = "431A2CF1683AAFAC";
            axMPComm1.Terminal2 = 8;
            axMPComm1.Terminal1 = 4;
            axMPComm1.DebugMode = false;

            // Open RS232
            axMPComm1.Port = "COM8:";
            axMPComm1.Speed = "9600";
            axMPComm1.Protocolo = 0;     // 0 -> RS232/485, 1 -> TCP/IP
            axMPComm1.RS485 = false;

            // Open Comm
            bool ret =axMPComm1.OpenComm();

            // RS232 successfully open
            if (ret)
            {
                // Target device
                axMPComm1.Pantalla = 1;         // Device with ID 1
                axMPComm1.BufferTX = (char) 0xE0 + "MP Test"; // Mode
'Scroll' + 'MP test'
                axMPComm1.Orden = 0x27;           // Send
                // Receive (if any)
                String s = axMPComm1.BufferRx;    // Get answer (if any)
             // Error
              int error = axMPComm1.ErrorNx;              // Get error number
             // Close Comm
             axMPComm1.CloseComm();
            }
        }
    }
```

Example 2: serial and Ethernet networks

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsDemoMPCOMMOCX
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      public void Test()
      {
         // Configure component
         Commom_Ini();


         // Test RS232_485
         RS232_485Version();

         // Test Ethernet TCP_IP
         TCP_IPVersion();
      }

      public void Commom_Ini()
      {
         // Configure component
         axMPComm1.LicProduct = "MPCOMM";
         axMPComm1.LicName = "PIXELMP";
         axMPComm1.LicKey = "431A2CF1683AAFAC";
         axMPComm1.Terminal2 = 8;
         axMPComm1.Terminal1 = 4;
         axMPComm1.DebugMode = false;
      }

      public void Common_Send(String s1)
      {
         // Open Comm
         bool ret = axMPComm1.OpenComm();

         // RS232 successfully open
         if (ret)
         {
```

```
            // Target device

            axMPComm1.Pantalla = 1;                       // Device with ID 1
            axMPComm1.BufferTX = (char)0x9A + 0xE4 + s1;          // Mode 'Scroll' + 'MP test'
            axMPComm1.Orden = 0x27;                        // Send

            // Receive (if any)
            String s = axMPComm1.BufferRx;                 // Get answer (if any)

            // Error
            int error = axMPComm1.ErrorNx;                 // Get error number

            // Close Comm
            axMPComm1.CloseComm();
        }
    }

    public void RS232_485Version()
    {
        // Open RS232
        axMPComm1.Port = "COM2:";
        axMPComm1.Speed = "9600";
        axMPComm1.Protocolo = 0;              // 0 -> RS232/485, 1 -> TCP/IP
        axMPComm1.RS485 = false;

        // Open Comm
        Common_Send("MP Test RS232485");
    }

    public void TCP_IPVersion()
    {
        // Open RS232
        axMPComm1.TipoServidor = 1;
        axMPComm1.HostPort = 53;
        axMPComm1.Usuario = "admin";
        axMPComm1.Password = "security";
        axMPComm1.HostAdress = "192.168.1.44";

        axMPComm1.Protocolo = 1;


        // Open Comm
        Common_Send("MP Test TCP IP");
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        Test();
       Close();
    }
  }
}
```